Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

**Ликино-Дулевский политехнический колледж – филиал ГГТУ**

**КУРСОВОЙ ПРОЕКТ**

Тема: «Разработка интернет-магазина по продаже велосипедов»

МДК.08.01 «Проектирование и разработка интерфейсов пользователя»

**Выполнил:**

Степанов Дмитрий Даниилович

Студент 3 курса группы ИСП.21.2А

09.02.07 Информационные системы и программирование)

очной формы обучения

**Руководитель:**

Гжегожевский Сергей Владимирович

Оценка

\_\_\_\_\_\_\_\_\_/\_\_\_­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Подпись руководителя

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ликино-Дулево

2024 год

**Введение**

Продажа — бизнес-понятие, описывающее практически любую коммерческую деятельность, бизнес вообще. Продажа чаще всего является завершающим этапом бизнес-цикла коммерческого предприятия. Употребляется всегда только в единственном числе.

Каждая современная, уважающая себя компания обязана иметь свой веб-магазин либо сайт, благодаря им она привлекает потенциальных покупателей к своей продукции и повышает доход из-за того, что людям даже не нужно выходить из дома для покупки чего-либо, все доставит курьер, либо вещь можно будет забрать прямо в магазине.

CycleStore это интернет магазин, он предназначен для ознакомления клиентов с компанией и ее продукцией, а так-же для продажи велосипедов. Сайт будет интересен как новичкам и любителям велоспорта, так и профи. На сайте можно создавать свой аккаунт для заказа велосипеда.

**1. Разработка системного проекта.**

**1.1. Назначение разработки**

Целью разработки является создание сайта для компании CycleStore, с возможностью просмотра продукции, добавления его в корзину. Также каждый пользователь должен создать свой аккаунт на сайте через регистрацию.

**1.2. Требования к программному продукту**

**1.2.1. Требования к функциональным характеристикам**

Для использования сайта подойдет любой браузер и понадобится доступ в Интернет

Функции:

1. Регистрация
2. Авторизация
3. Добавление/удаление/редактирование товара
4. Заказ товара
5. Поиск товара

**1.2.2. Требования к надежности и безопасности**

Для покупок нужно зарегистрировать/войти в аккаунт

Для регистрации пользователь должен ввести желаемый логин, электронную почту и пароль с его дубликатом. Для безопасности, к паролю выдвигаются следующие требования:

1. Он должен содержать минимум 8 символов
2. Он не должен быть простым или распространенным
3. Он не может состоять из одних цифр

Пароль после регистрации попадает в базу и будет хешироваться, благодаря чему невозможно украсть пароль.

**1.2.3. Требования к составу и параметрам технических средств**

Для разработки использовалась программа PyCharm, т.к. является наиболее удобной для работы с проектом.

**2. Разработка технического проекта**

**2.1. Обоснование выбора CASE – средств**

**2.2. Проектирование модели данных**

**2.2.1. Диаграмма прецедентов**

Администратор – управляет сайтом, добавляет/редактирует/удаляет товары из каталога, отслеживает поступающие заказы.

Пользователь – регистрирует и входит в аккаунт, просматривает каталог, добавляет товары в корзину, заказывает товары.

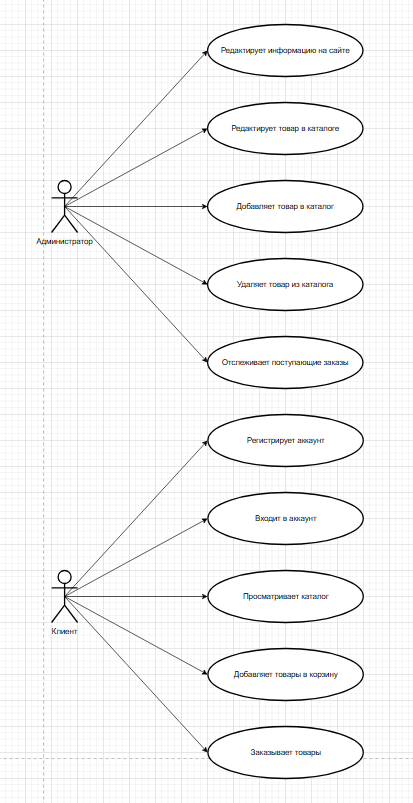


Рис. . «»

**2.2.2. Моделирование бизнес-процессов**

Таблица. . «»

|  |  |
| --- | --- |
| **Номер бизнес-процесса** | **Название бизнес-процесса** |
| 1-Регистрация\_Пользователя | Регистрация пользователя на сайте |
| 2-Вход\_Пользователя | Вход пользователя в аккаунт на сайте |
| 3-Добавление\_Товара | Добавление товара в каталог |
| 4-Заказ\_Товара | Заказ товара |
| 5-Удаление\_Товара | Удаление товара |
| 6-Редактирование\_Товара | Редактирование товара |

**2.2.3. Словесный алгоритм бизнес-процессов**

Процесс покупки товара:

1. Администратор размещает товар в каталоге
2. Клиент заказывает товар из каталога
3. Администратор принимает заказ

**2.2.4. Построение диаграммы действий**

Таблица. . «»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Диаграмма и номер операции | Операция | Исполнитель | Как часто | Входящие документы | Исходящий документ |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1-Заказ\_  Товара | Выставляет товар в каталог | Администратор | По плану | Каталог | Нет |
| 2-Заказ\_  Товара | Заказывает товар из каталога | Клиент | По мере необходимости | Каталог | Нет |
| 3-Заказ\_  Товара | Принимает заказ | Администратор | Обязательно | Нет | Нет |

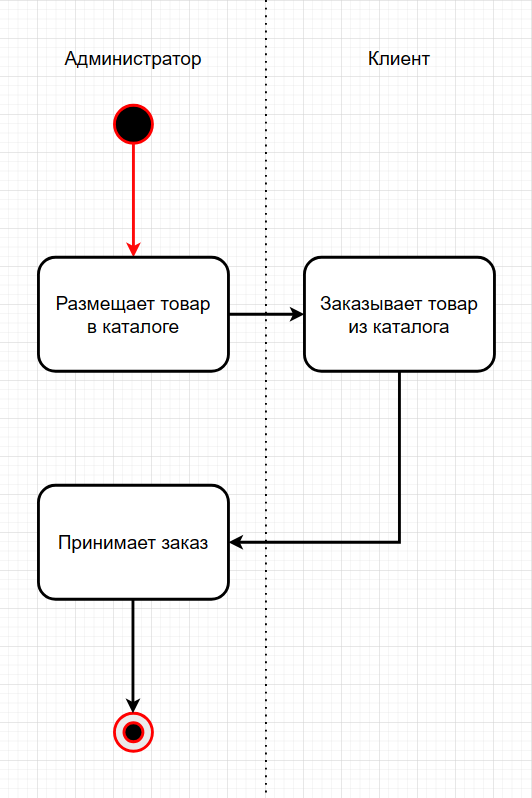
****

Рис. . «»

**3. Реализация приложения**

**3.1. Обоснование выбора средств разработки**

В рамках курсового проекта было принято решение разработать сайт, сайт по продаже велосипедов. Учитывая возможности, оборудования и программного обеспечения, необходимо создать современный программный продукт, избегая таких недостатков существующих коммерческих предложений, как высокая стоимость внедрения и сопровождения и слабая ориентированность на пользователя с разной профессиональной подготовкой. Также необходимо уделить особое внимание надежности приложения и простоте его интерфейса.

Поэтому для разработки автоматизированной системы учета продукции были выбраны язык программирования Python и программа PyCharm.

**3.2. Руководство программиста**

* + 1. **Построение моделей и структура БД.**

Корзина (cart)

from django.db import models  
from django.contrib.auth.models import User  
from main.models import Tovar  
  
class CartItem(models.Model):  
 tovar = models.ForeignKey(Tovar, on\_delete=models.CASCADE)  
 quantity = models.PositiveIntegerField(default=0)  
 user = models.ForeignKey(User, on\_delete=models.CASCADE)  
 date\_added = models.DateTimeField(auto\_now\_add=True)  
   
 def \_\_str\_\_(self):  
 return f'{self.quantity} x {self.tovar.cyclename}'

main

class Tovar(models.Model):  
 cyclename = models.CharField(max\_length=100, blank=False, default="")  
 tovarprice = models.FloatField()  
 category = models.ForeignKey(  
 "Categories",  
 related\_name="tovars\_categories",  
 on\_delete=models.CASCADE,  
 null=True,  
 )  
 shipping = models.TextField(max\_length=300, blank=False, default="")  
 tovardescrpt = models.TextField(max\_length=500, blank=False, default="")  
 date\_posted = models.DateTimeField(default=timezone.now, null=True)  
 tovarimage = models.ImageField(  
 upload\_to="img/tovarimages",  
 verbose\_name="Фото товара",  
 default=None,  
 blank=False,  
 )  
  
 def \_\_str\_\_(self):  
 return f"{self.cyclename} {self.category}"  
  
  
  
class Post(models.Model):  
 title = models.CharField(max\_length=100, blank=False)  
 postbody = models.TextField(max\_length=1000)  
  
 def \_\_str\_\_(self):  
 return self.title

**3.2.2. Представления**

Представление cart

В корзину добавляются товары, которые можно заказать и удалить из нее.

def view\_cart(request):  
 if request.user.is\_authenticated:  
 cart\_items = CartItem.objects.filter(user=request.user)  
 total\_price = sum(item.tovar.tovarprice \* item.quantity for item in cart\_items)  
 return render(  
 request,  
 "cart/cart.html",  
 {"cart\_items": cart\_items, "total\_price": total\_price},  
 )  
 else:  
 return render(request, "account/signup.html")

def add\_to\_cart(request, tovar\_id):  
 tovar = Tovar.objects.get(id=tovar\_id)  
 cart\_item, created = CartItem.objects.get\_or\_create(tovar=tovar, user=request.user)  
 cart\_item.quantity += 1  
 cart\_item.save()  
 return redirect("cart:view\_cart")  
  
  
def remove\_from\_cart(request, item\_id):  
 cart\_item = CartItem.objects.get(id=item\_id)  
 cart\_item.delete()  
 return redirect("cart:view\_cart")

Представление orders

В заказах отображаются заказанные товары.

def order\_create(request):  
 cart = CartItem.objects.all()  
 if request.method == 'POST':  
 form = OrderCreateForm(request.POST)  
 if form.is\_valid():  
 order = form.save()  
 for item in cart:  
 OrderItem.objects.create(order=order,  
 tovar=item['tovar'],  
 price=item['price'],  
 quantity=item['quantity'])  
 # очистка корзины  
 cart.clear()  
 return render(request, 'orders/order-created.html',  
 {'order': order})  
 else:  
 form = OrderCreateForm  
 return render(request, 'orders/create-order.html',  
 {'cart': cart, 'form': form})

Представление Index

На главной странице есть блок с товарами (каталог)

def index(request):  
 tovars\_all = (Tovar.objects.all(),)  
 posts = Post.objects.all()  
 filter\_tovar = TovarFilter(request.GET, queryset=Tovar.objects.all())  
 return render(request, "main/index.html", {"tovars": filter\_tovar, "posts": posts})

**3.2.3. Шаблоны**

Создан шаблон «base», он имеет свои стили и благодаря нему все страницы имеют одинаковый внешний вид, из-за этого страницы не будут выделяться между собой. Этот шаблон подключен ко всем страницам.

{% load static %}  
<!DOCTYPE html>  
<html lang="ru">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <link rel="stylesheet" href="{% static 'css/styles.css'%}" type="text/css">  
 <title>cyclestore | {% block title%}{% endblock %}</title>  
</head>  
  
<body>  
 <img src="{% static 'img/.gif' %}" alt="" id="">  
 <header>  
 <div class="header-wrapper">  
 <div class="logo"><a href="{% url 'index' %}"><img src="{% static 'img/logo.png' %}" alt="logo"></a></div>  
 {% if user.is\_authenticated %}  
 <form action="{% url 'create-tovar' %}" target="\_blank">  
 <button id="createOB">Добавить товар</button>  
 </form>  
 {% else %}  
 {% endif %}  
 <div id="userlog">  
 {% if user.is\_authenticated %}  
 {{ user.username }}  
 <a class="main-header\_links" href="{% url 'cart:view\_cart' %}" id="avatar"><img  
 src="{% static 'img/cart\_l.png' %}" alt="avatar"></a> |  
 <a class="main-header\_links" href="{% url 'account\_logout' %}">Выйти</a>  
 {% else %}  
 <a class="main-header\_links" href="{% url 'account\_login' %}">Войти</a> / <a  
 href="{% url 'account\_signup' %}" class="main-header\_links">Зарегестрироваться</a>  
 {% endif %}  
 </div>  
 </div>  
 </header>  
 <div class="wrapper">  
 <!-- навигация -->  
 <div class="main-nav">  
 <li><a href="{% url 'index' %}">Главная</a></li>  
 <li><a href="{% url 'about' %}">О нас</a></li>  
 <li><a href="{% url 'services' %}">Полезная информация</a></li>  
 </div>  
 <!-- контент -->  
 <div class="container">  
 <div class="content">  
 {% if messages %}  
 {% for message in messages %}  
 <div class="alert alert-{{ message.tags }}">  
 {{ message }}  
 </div>  
 {% endfor %}  
 {% endif %}  
 {% block content %}{% endblock %}  
 </div>  
 </div>  
 </div>  
 <footer>  
 <div class="footer-wrapper">  
 <div class="footer-grid">  
 <div class="footer\_\_c">  
 <p>(с) CycleStore <br>  
 Все права защищены</p>  
 </div>  
 <div class="footer\_\_socials">  
 <a href=""><img src="{% static 'img/socials\_vk.png' %}" alt="social-1"></a>  
 <a href=""><img src="{% static 'img/socials\_ok.png' %}" alt="social-2"></a>  
 <a href=""><img src="{% static 'img/socials\_tg.png' %}" alt="social-3"></a>  
 </div>  
 <div class="footer\_contacts">  
 <p><span id="footer\_\_phone">+ 7 968 714 4502</span> - 09:00 - 20:00 ПН-ВС</p>  
 <p id="footer\_\_email">cycle-store@mail.ru</p>  
 </div>  
 </div>  
 </div>  
 </footer>  
  
</body>  
  
</html>

**3.2.4. Формы**

Форма добавления товара

class TovarForm(ModelForm):  
 def \_\_init\_\_(self,\*args,\*\*kwargs):  
 super().\_\_init\_\_(\*args,\*\*kwargs)  
 self.fields['category'].empty\_label = 'Категория не выбрана'  
   
 category = forms.ModelChoiceField(queryset=Categories.objects.all(), empty\_label="Категория", required=True)  
 class Meta:  
 model = Tovar  
 fields = ["tovarname", 'tovarprice', 'category', 'shipping', 'tovardescrpt','tovarimage']  
 widgets = {  
 "tovarname": TextInput(attrs={'placeholder': 'Заголовок товара', 'class': 'form-input tovarname-input'}),  
 "tovarprice": TextInput(attrs={'placeholder': 'Цена', 'class': 'form-input'}),  
 "category": forms.Select(attrs={'placeholder': 'Категория', 'class': 'form-input'}),  
 "shipping": Textarea(attrs={'placeholder': 'Доставка', 'class': 'form-input'}),  
 "tovardescrpt": Textarea(attrs={'placeholder': 'Опишите хаарктеристики товара, его состояние, нюансы', 'class': 'form-input'}),  
 }

**3.3. Руководство пользователя**

Главная страница. Это первая страница, на которую попадает пользователь, входя на сайт. Здесь находится каталог (список со всеми товарами магазина, которые есть на данный момент), поисковая строка, блок с кнопками, который переносит на страницы Главная / О нас / Сервис, а также кнопки, которые перебрасывают на формы авторизации и регистрации.

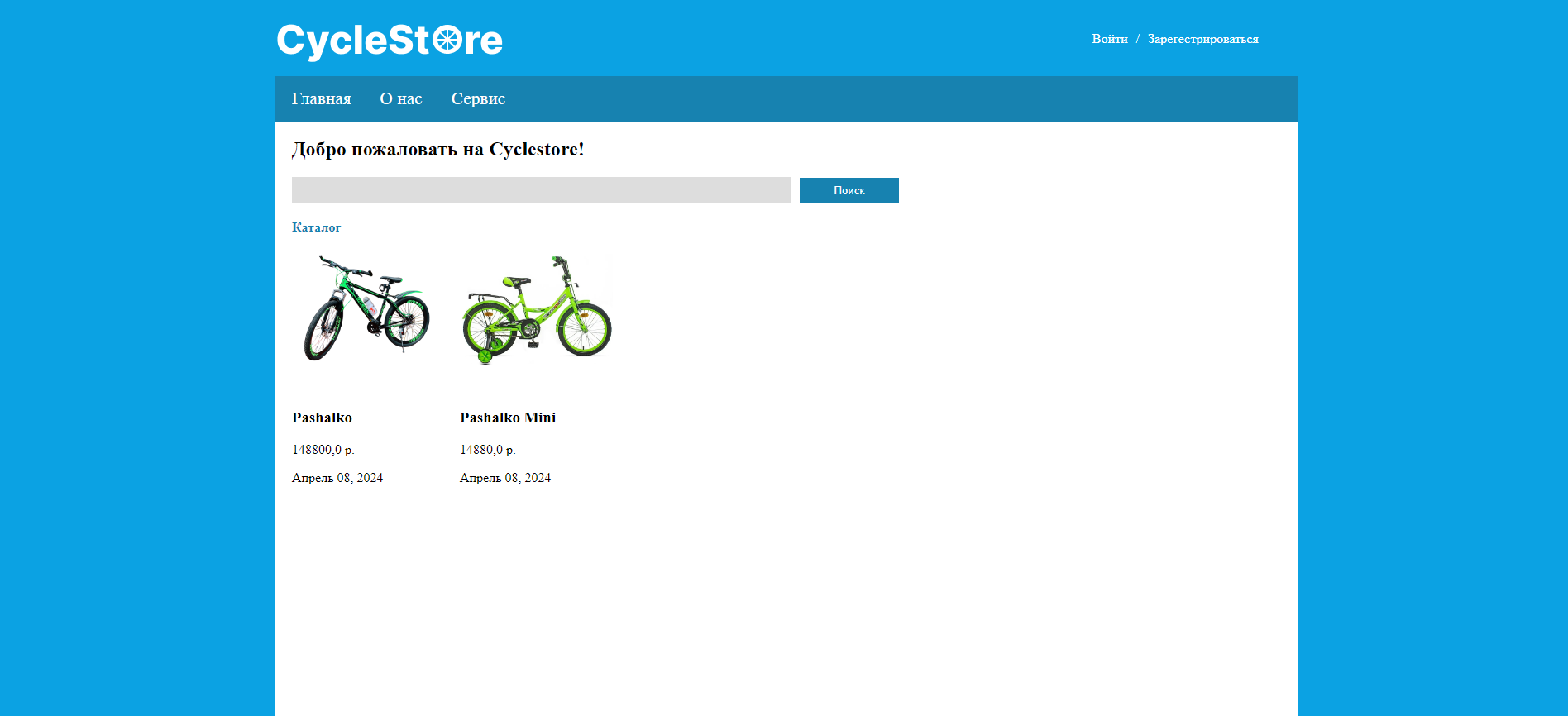


Рис. . «»

О нас. Эта страница содержит информацию о компании и ее логотип.

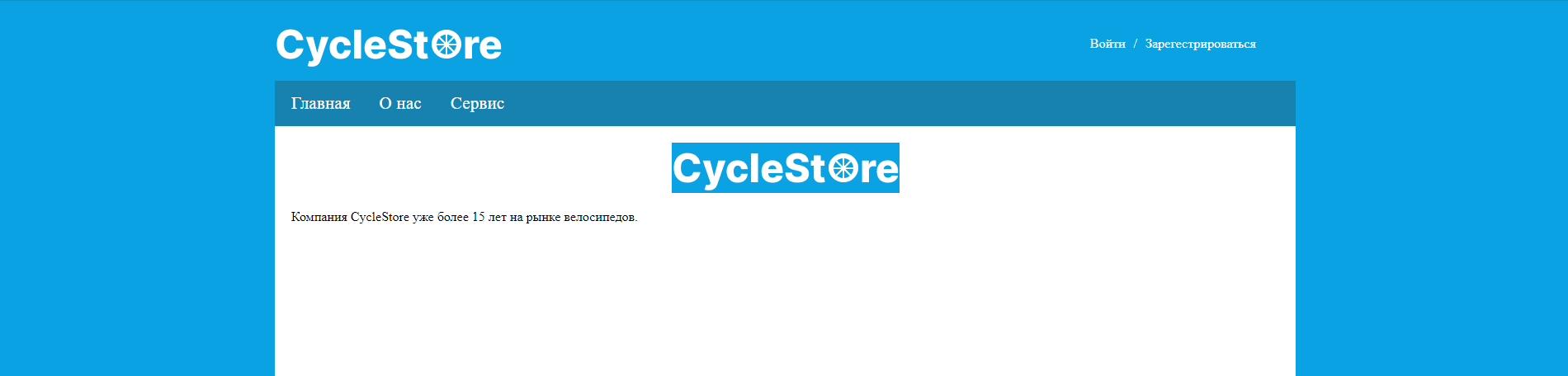


Рис. . «»

Сервис. На этой странице содержится информация, которая касается обслуживания велосипедов в мастерских компании.

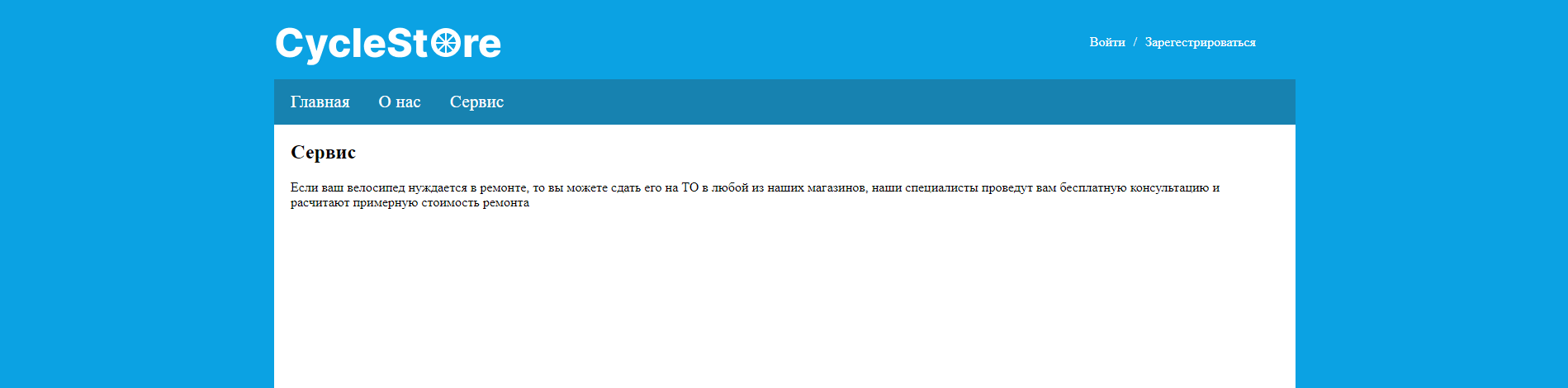


Рис. . «»

Вход в аккаунт. Для входа в аккаунт на сайте нужно войти в аккаунт через кнопку «Войти» и заполнить форму со своими данными.

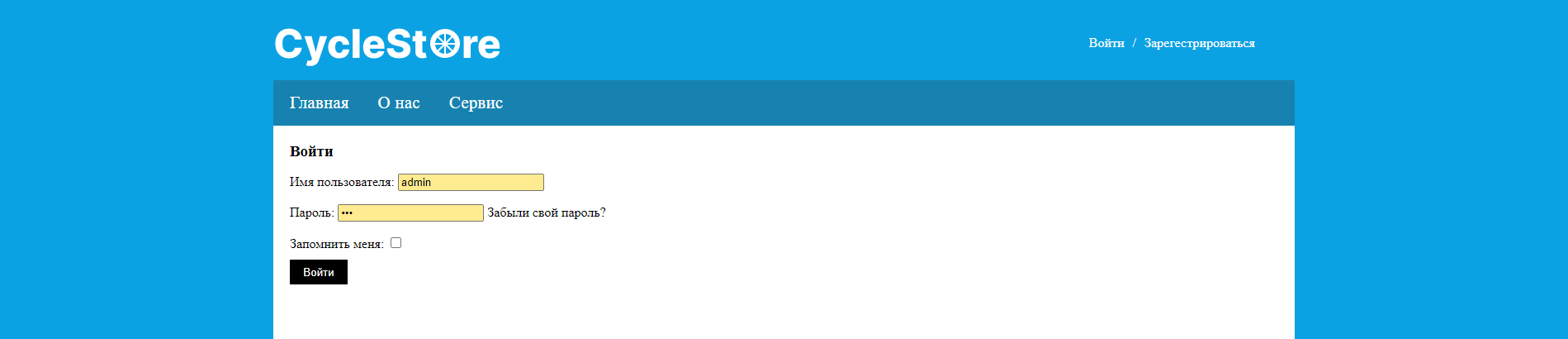


Рис. . «»

Регистрация аккаунта. Для регистрации на сайт нужно зарегистрировать в аккаунт через кнопку «Зарегистрироваться» и заполнить форму со своими данными.

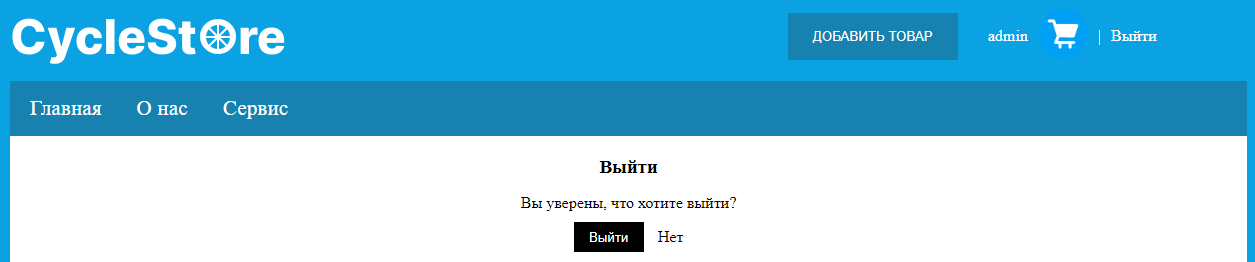


Рис. . «»

Выход из аккаунта. Что-бы выйти из аккаунта нужно нажать кнопку «Выйти» и подтвердить действие

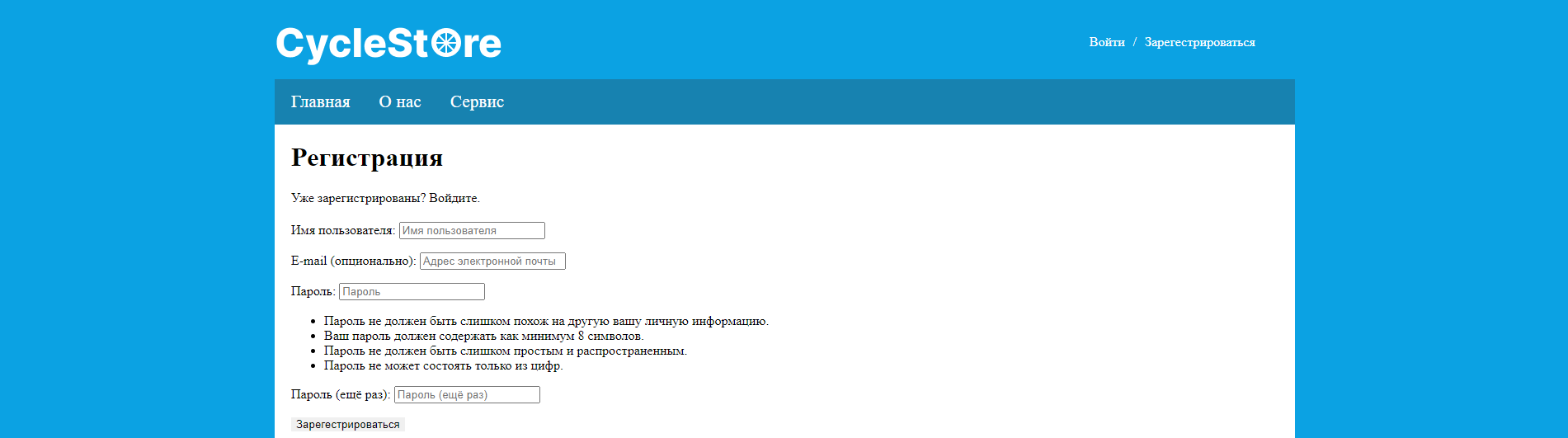


Рис. . «»

Добавление товара. Тут нужно заполнить все формы, добавить картинку и нажать кнопку добавить. Тоже самое можно сделать через админ-панель. Удаление товара происходит только через админ-панель

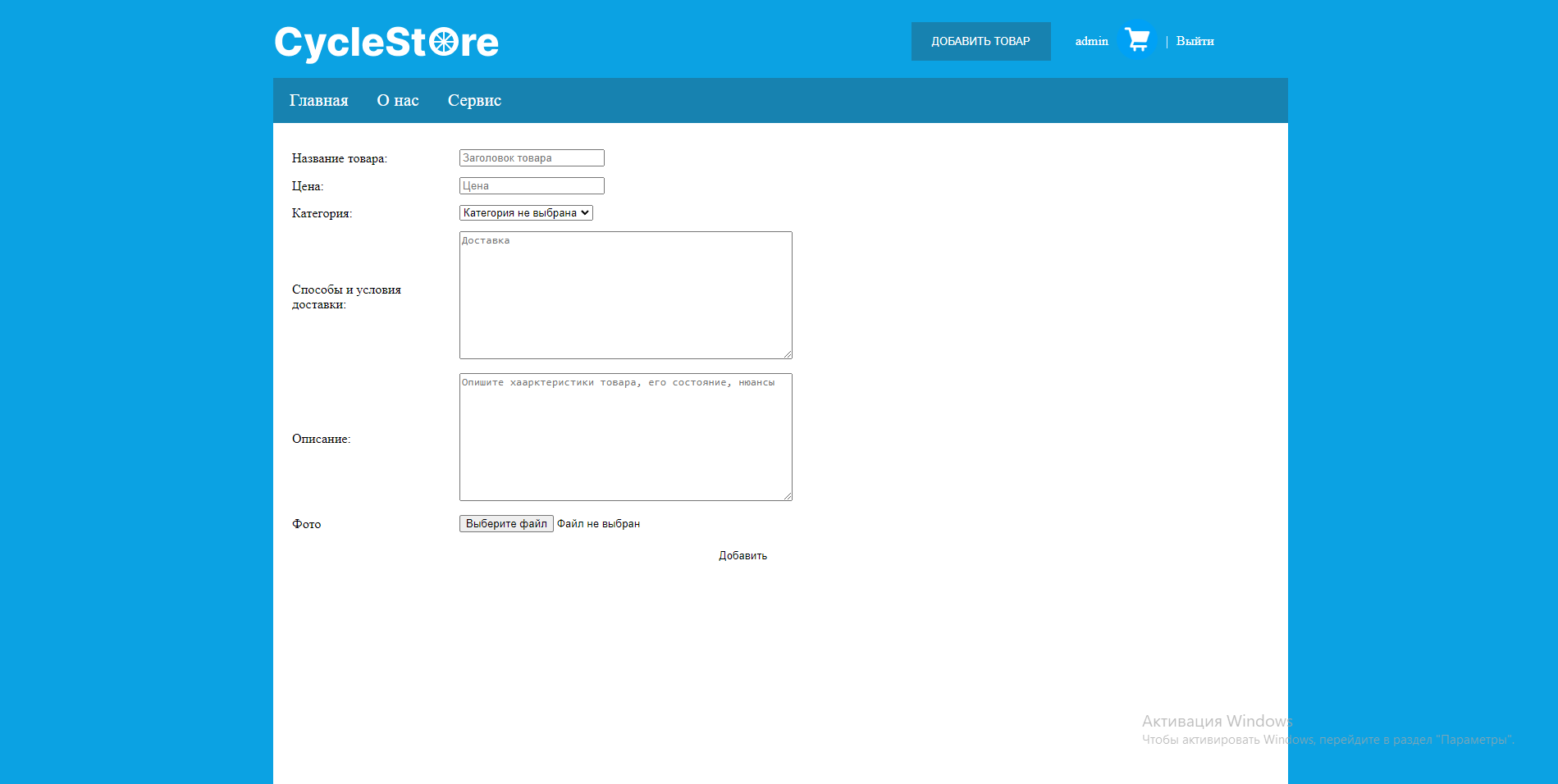


Рис. . «»

Страница товара. На ней располагается фотография, описание, доставка и название товара. Для заказа товара нужно нажать на кнопку «Добавить в корзину».

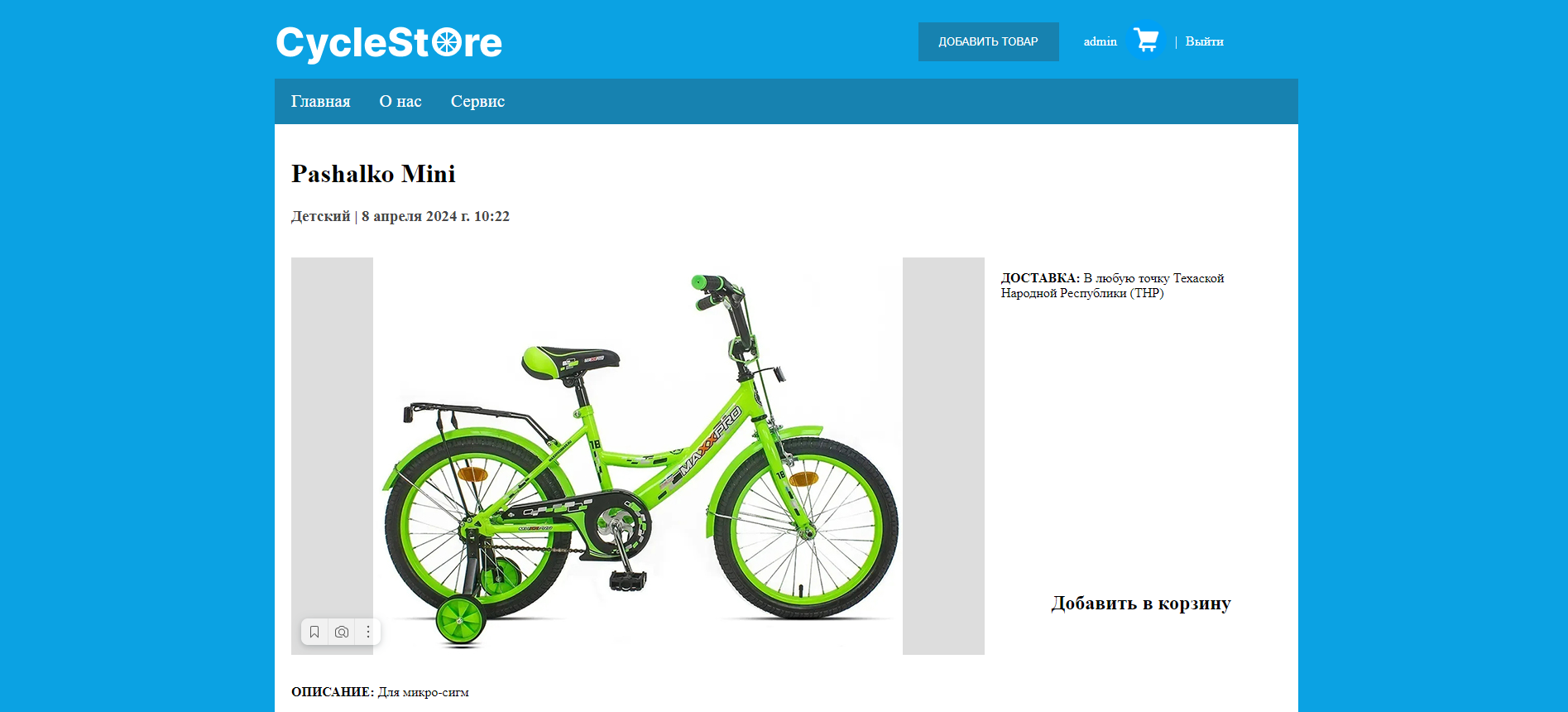


Рис. . «»

Корзина. Для перехода в нее требуется нажать на значок корзины. В ней можно убрать товар из корзины, либо заказать его. Для оформления товара требуется нажать кнопку «Создать заказ».

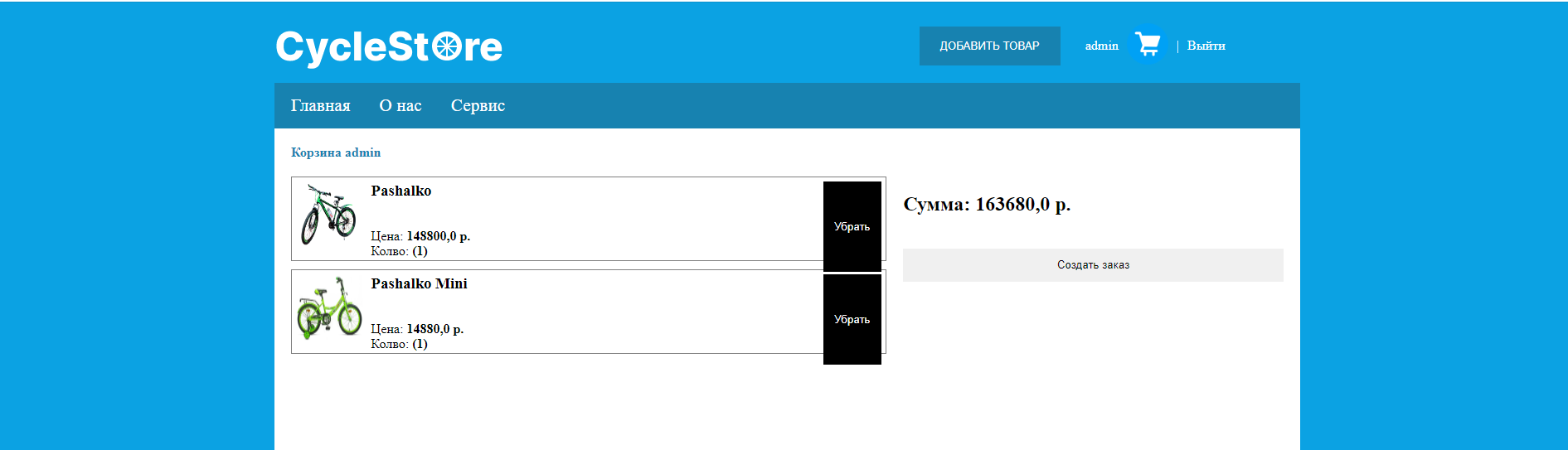
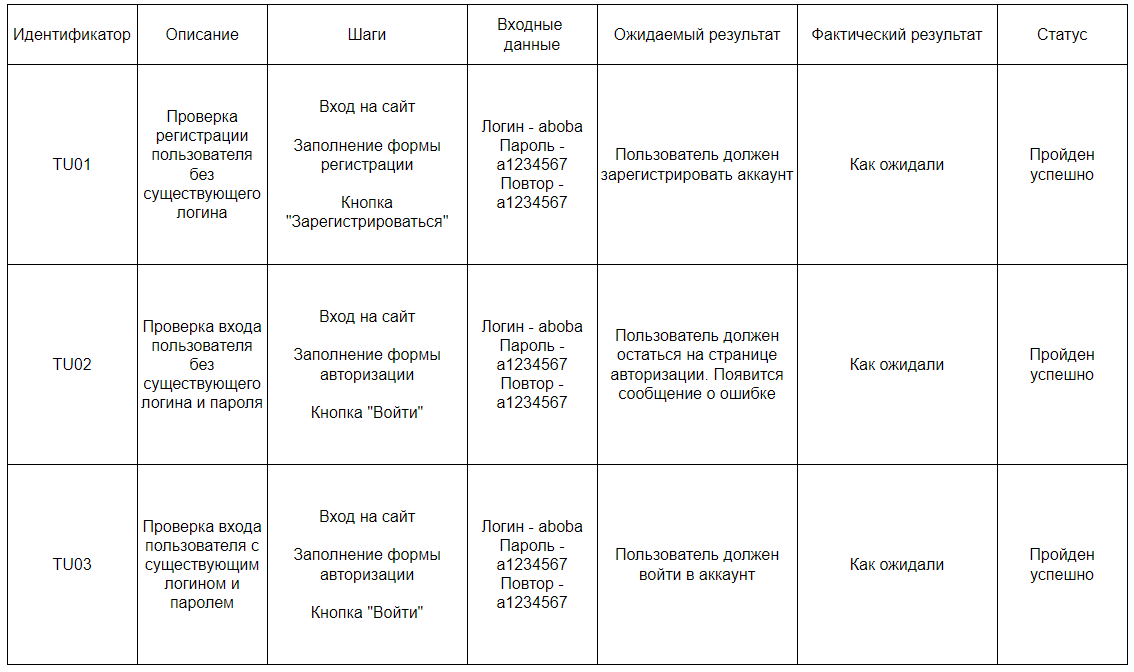


Рис. . «»

**4. Тестирование программного продукта**

**4.1. Тестовые сценарии**

Таблица. . «»



TU01 – Проверка регистрации пользователя без существующего логина:

Клиент должен зайти на сайт, далее перейти на страницу регистрации на сайте, ввести желаемый логин, пароль и дубликат пароля, а так-же почту при желании, далее нажать на кнопку «Зарегистрироваться». После регистрации пользователя перенесет на главную страницу.

TU02 – Проверка авторизации пользователя без существующего логина:

Клиент должен зайти на сайт, далее перейти на страницу авторизации на сайте, ввести свой логин и пароль, далее нажать на кнопку «Войти». После этого система оставит его на этой странице и выдаст ошибку.

TU03 – Проверка авторизации пользователя без существующего логина:

Клиент должен зайти на сайт, далее перейти на страницу авторизации на сайте, ввести свой логин и пароль, далее нажать на кнопку «Войти». После авторизации пользователя перенесет на главную страницу.

**4.2. Юнит-тесты в Django**

**5. Размещение проекта на хостинге.**

Регистрация на хостинге

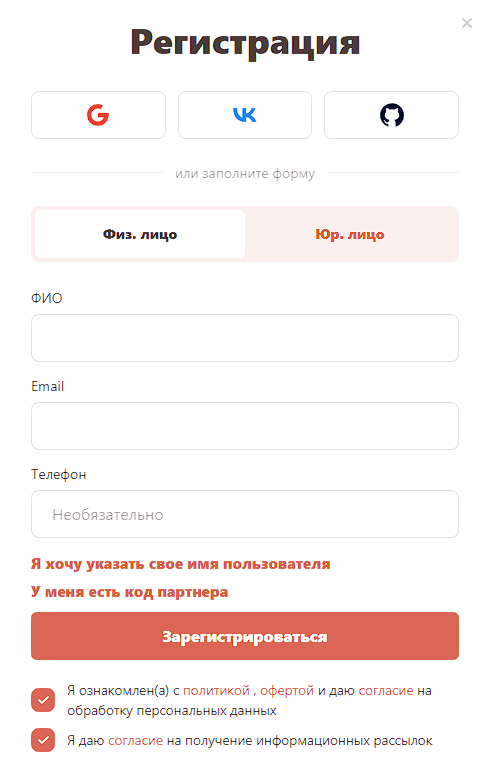


Рис. . «»

В списке доменов у вас должен появиться домен

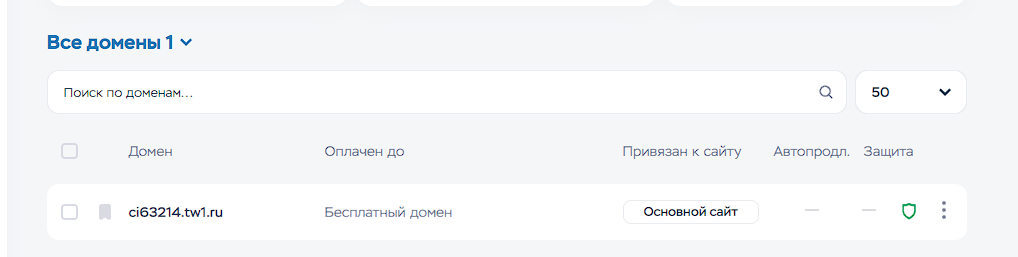


Рис. . «»

Перейдите на Главную страницу и включите доступ по SSH

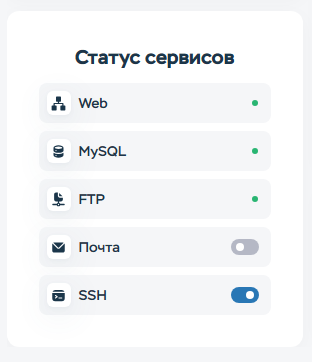


Рис. . «»

Скачайте программу putty

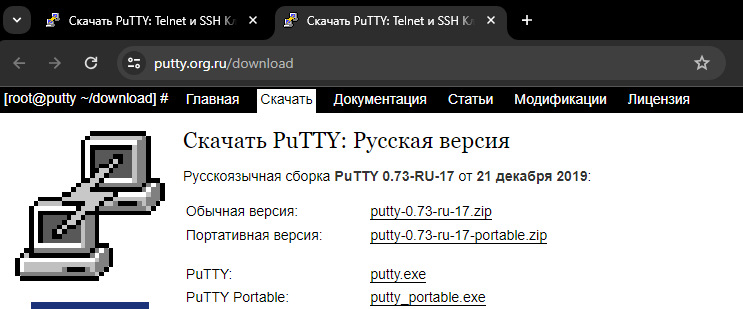


Рис. . «»

Запустите программу

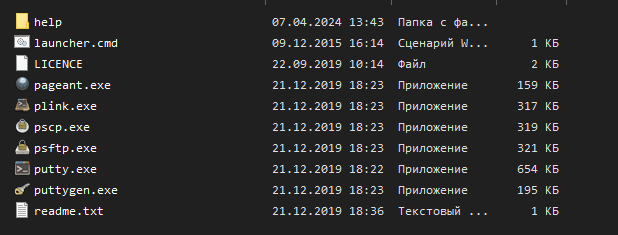


Рис. . «»

В подключении укажите домен

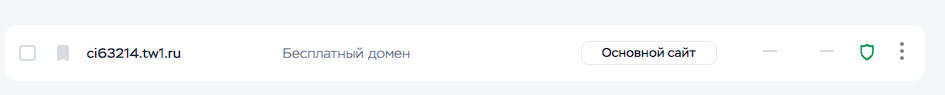


Рис. . «»

Затем введите логин и пароль в командную строку, которые пришли в СМС от сайта.

**Заключение**

В данной курсовой работе был создан сайт «CycleStore». Для его разработки я использовал заранее разработанный дизайн. Проект создавался в среде PyCharm на основе Django. В результате выполнения работы по разработке сайта были использованы CASE-средства для создания диаграмм прецедентов и действий, построены представления о вёрстке проектор, сформированы функциональная и каркасная архитектура сайта.

Список литературы